Neural Computing and Applications (2022) 34:15499–15512
https://doi.org/10.1007/s00521-022-07253-x

ORIGINAL ARTICLE

A low latency traffic sign detection model with an automatic data labeling pipeline

Jiapeng Luo[1] · Zhongfeng Wang[1]

Received: 18 October 2021 / Accepted: 29 March 2022 / Published online: 23 April 2022

Abstract
As a crucial sub-task for autonomous driving and intelligent transportation, traffic sign detection attracts a lot of researchers' attention. Although prior works have achieved promising results, it still suffers from two problems, the need of massive labeled data and the slow inference speed on high-resolution images, which are also the common problem of generic object detection. Motivated by these problems, we propose a lightweight traffic sign detection model and employ an automatic data labeling pipeline. The detection model utilizes a cascaded structure that consists of a localization module and a recognition module, achieving quite fast speed on GPU and edge devices. We discard all complex structures and operations to boost the speed of the model, making it easy for deployment. Then, we propose a two-stage automatic data labeling pipeline to reduce the cost of data labeling work. With only traffic sign template images, a synthetic dataset is constructed for generating initial pseudo labels in the first stage. In the second stage, we propose to use an image pretext model to refine the initial labels. The accuracy of the final pseudo labels is nearly 100%. We test the proposed method on TT-100K, GTSDB, and GTSRB datasets, and the results show that the model trained with the pseudo labels only has a negligible accuracy loss compared with the model trained by real labels. The proposed model's calculation latency is around 1 ms on GPU, and the accuracy is still on par with the state-of-the-art models.

Keywords Traffic sign detection · Image clustering · Real-time detection · Automatic labeling · Intelligent transportation

1 Introduction

Traffic sign detection and recognition (TSD and TSR) play an important role in many transportation-related applications, such as autonomous driving systems and intelligent transportation systems. The targets of TSD and TSR are locating the presence of traffic signs with a bounding box and identifying its types in the box. For autonomous driving systems, it is an essential way to sense the traffic limits and other crucial information from traffic signs. For driving assistant systems, it is a primary way to remind the human driver of the important road information to increase driving safety [29]. In some prior works, the term "traffic sign detection" is limited to the task of localization of traffic signs specifically, while "recognition" refers to classifying the traffic signs. But for the generic object detection task, the detection includes both localization and recognition. In the rest of this paper, we follow the convention of generic object detection that TSD includes localization and recognition tasks.

TSD has been widely researched in recent years. Similar to the generic object detection task, the classical approaches for TSD focus on handcrafted features such as color, texture, edge, and other low-level features, where a classifier is employed to classify the features in a sliding window [19, 29]. These approaches fail when the traffic sign appearance is changed drastically by light, pose, motion blur, etc. Nowadays, the convolutional neural network (CNN) has achieved great success in visual tasks, and it has become the most popular detection and recognition method. We give a brief review of TSD in Sect. 2. The CNN-based approaches require collecting and annotating a

✉ Zhongfeng Wang
zfwang@nju.edu.cn

Jiapeng Luo
luojiapeng@smail.nju.edu.cn

[1] School of Electronic Science and Engineering, Nanjing University, No. 163 Xianlin Road, Nanjing 210000, Jiangsu, China

large number of images for network training. With the massive annotated data, the CNN's weights are updated by a back-propagated gradient algorithm to learn better visual features automatically. By training with large-scale labeled datasets, state-of-art CNN-based object detectors can achieve very high detection accuracy.

However, there are still several problems with TSD. First, the prior detection framework is not efficient enough for TSD. Traffic signs are usually very small because of the long distance to the camera. The proportion of a traffic sign in an image is typically around 1.6×10^{-4}, where the sign has 25×25 pixels, and the image has 2000×2000 pixels, as shown in Fig. 9a. The generic object detection often uses a smaller input resolution, such as 640 pixels and 320 pixels [34], which increases the false negative rate of small signs. It is not computational economic to directly expand the input resolution, because the computation cost of a CNN is roughly square related to the input image size, which is also a considerable and essential factor for deployments to mobile devices. The second troublesome problem is that training a CNN-based method needs massive annotated data. Since labeling such an amount of data is expensive and time-consuming, it is also a common barrier to deep learning. Furthermore, the traffic signs on roads often update and change, which requires continued new annotation data to keep the model accurate. The traffic sign types are also various which are usually more than 40 in public datasets and more than 100 in the real road. It imposes extra loads on labeling jobs. There have already been unsupervised and semi-supervised methods, such as SCAN [46] and SPICE [31], focusing on image clustering with no label or few labels. However, most of them are only suitable for small-scale datasets like CIFAR [21] and STL10 [10], and fail on the real scenario datasets of traffic signs because of the serious class imbalance problem.

Motivated by the above problems, we propose an efficient traffic sign detector and an automatic data labeling pipeline. First, we design an efficient cascaded detection framework composed of a localization module and a recognition module. The localization module is a single-class object detector based on a lightweight network. For fast inference speed and easy deployment, the network only contains regular convolution without other complex operations and structures. According to the predicted location of traffic signs, we crop the image patches from the original image and use an efficient image classification network to predict the type of these traffic signs. In generic object detection, context information around the object is essential for small and difficult objects. However, the context pixel only provides very limited information for the traffic sign recognition process. The benefit of separating the localization task and classification task is that the detection module can focus on finding the traffic signs ignoring the types and vice versa for the classification module. It is more efficient for TSD tasks. The proposed lightweight detector is extremely fast on high-performance GPU and edge devices, which is very important for deployment environments.

To reduce the workload of manual annotation for the recognition task, we propose an automatic data annotation pipeline. The proposed pipeline consists of two stages and only needs a template image of each traffic sign type as the initial information. In the first stage, we construct a synthetic dataset making use of template images and various visual transformations. A classification model is trained with the synthetic dataset and is employed to predict the initial pseudo labels. Although the visual transformation is various during synthetic dataset construction, the data distribution of the synthetic datasets is still inevitably distinct from the real dataset. Therefore, the accuracy of the initial pseudo labels is not satisfactory. In the second stage, to enhance the accuracy of the pseudo labels, we refine pseudo labels by a visual pretext model. The pretext model projects the input image to the latent space where similar images tend to be closer to each other. The samples with the same pseudo labels but at a large distance are identified as an unreliable label. A better classifier can be trained with the refined pseudo labels.

The major advantages of the proposed approaches can be concluded as two points. First, the detection framework is very efficient. It does not introduce any other complex operations and is easy to deploy. The required computation resource is small too. Second, benefiting from the automatic labeling pipeline, we do not need to provide the explicit types of each traffic sign when annotating the datasets. Only the traffic signs bounding boxes are required during the training process, which greatly reduces the annotation workloads.

To demonstrate the effectiveness of our approach, the proposed model is evaluated on three popular public datasets, TT-100K [55], GTSDB [19], and GTSRB [40]. We give a comprehensive comparison between our approach and prior methods. Specifically, on TT-100K, the proposed detection pipeline achieves the highest mean average precision (mAP) using the original annotation data. With high accuracy pseudo labels, the automatically annotated data only has a negligible accuracy loss. The inference speed of the proposed model is also very fast. Ignoring the latency of data transmission, the average computation latency of the proposed model is around 1 millisecond on GPU. We also evaluate the proposed model on an edge device, which shows it can achieve fast inference speed, and the average computation latency is around 6 milliseconds.

The major contributions of this paper are summarized as follows: (1) we design an efficient traffic sign detector that consists of a localization module and a recognition module with millisecond-level latency. (2) We propose an automatic labeling pipeline to significantly reduce the manual annotation work. (3) Extensive experiments are conducted, and the results show that the proposed TSD method is suitable for real applications in terms of accuracy and computation cost.

2 Related work

There are already many prior works focusing on object detection and unsupervised learning. However, most of these methods have computing resources limitations and complex network structure, which make it difficult to be applied to deployment environments immediately. In this section, we give a brief introduction to TSD, image clustering, and visual representation learning which are closely related to the proposed method.

2.1 Traffic sign detection

TSD is a specific case of generic object detection. The generic object detection techniques can certainly be applied directly to TSD, but the distinguishing characteristics of the traffic sign data make it not the best way for TSD. For example, traffic signs only occupy a few proportions of an image, and a series of down-sampling layers in generic object detection makes it difficult to obtain good features.

The characteristics of the classical works is to find efficient handcrafted image features, such as color, shape, and edge features [12, 28, 36, 37]. Berkaya et al. [2] utilized multi-features with an SVM classifier to detect traffic signs. Li et al. [23] integrated image segmentation with color invariants and the shape matching with the pyramid histogram of oriented gradient features. Although the classical methods do not need massive annotation data, they are vulnerable to diverse environmental disturbances, such as illumination, pose, motion blur, etc.

With the development of deep learning, CNN-based methods have developed rapidly in recent years and have become the most popular approach. To improve the detection performance for traffic signs, most prior CNN-based works choose to propose various customized designs based on the generic object detection models. Based on DSSD [13], Tian et al. [45] used the recurrent attention mechanism to fuse the information of neighboring receptive fields. With a hinge loss stochastic gradient descent, Jin et al. [20] improved the TSR performance in the training procedure. Shen et al. [39] proposed an adaptive pyramid convolution module, which makes features of different scales be adaptively fused and can be repetitiously optimized through back-propagation. Besides one-stage detectors, the cascaded detection framework is also an efficient choice. Wu et al. [47] utilized a two-level framework composed of a region proposal module and a classification module. A data augmentation method based on traffic sign logos is employed to enhance the diversity of training samples. Zhang et al. [52] proposed a cascaded R-CNN network to obtain the multiscale features in pyramids for small traffic sign instances. A data augmentation method is also proposed to increase the hard samples, which solves the data imbalance problem. PCN [24] is also a cascaded model containing two stages. By adopting a deep feature pyramid architecture with skip connections, PCN has more advantages for extracting features of small objects. Tabernik et al. [42] employ Mask RCNN to achieve good traffic sign detection performance with several improvements, such as data augmentation and hard sample mining. Besides the above TSD methods, there are also many works focusing on TSR [41, 51, 54]. For the sake of space, more details will not be described here.

There are already few weak/semi-supervised algorithms that can reduce the need of labeled data for traffic signs. ROSSOT [30] uses 60% labeled training samples and directly uses a classifier to generate pseudo labels for unlabeled samples. Then a better classifier is trained on the new data with pseudo labels. Multiple feature fusion (MFF) [17] uses a multiple features representation method to achieve better semi-supervised learning. We also provide a comparison between these two methods and the proposed method in Table 5. Besides weak/semi-supervised algorithms, some image augmentation methods have the potential to increase the accuracy of weak/semi-supervised algorithms. However, their original purpose is to increase the accuracy of the supervised algorithm and still need to utilize the full training datasets to achieve good performance. Their performances are not good with only part training data or synthetic data. So, we do not compare the proposed method with these methods for fairness. For interested readers, we list our searched augmentation methods as [3, 9, 14, 18].

2.2 Image clustering

Image clustering is a general technique the most related to our goals. Image clustering aims at grouping similar images into the same clusters without explicit annotations, which is an essential task in unsupervised learning. The measurement of similarity and discrepancy between samples is the core of clustering methods. Compared with classical clustering methods, unsupervised deep clustering shows significant superiority. Benefiting from the data-driven approach, deep clustering usually has a better image feature representation. The auto-encoder is firstly applied to image clustering [22, 44, 48, 53]. The stacked auto-encoders (SAE) use the encoder network to obtain the code of the input image and reconstruct the image by decoder network. It is trained to minimize the reconstruction error, so the code can preserve better information of the original image. Then the encoder can be combined with traditional clustering algorithms, such as k-means and spectral clustering. However, SAE-based methods tend to focus on low-level features because of the pixel-wise reconstruction loss. DeepCluster [4], DAC [6], and JULE [49] perform end-to-end learning pipelines, which train the CNN iteratively during clustering to improve the feature quality. But the accumulated errors can still damage the performance during iterations.

Visual representation learning has emerged as a popular method for image clustering. Compared with SAE-based generative models, contrastive learning-based methods, such as SimCLR [8], MOCO [16], SwAV [5], and BYOL [15] are discriminative approaches, which train the networks to perform image pretext tasks. They update the CNN by bringing representations of different views of the same image closer and spreading representations of views from different images apart, as the representations in the latent space are very suitable for image clustering. SCAN [46] employs the SimCLR to obtain semantically meaningful features and use it as a prior in a learnable clustering approach. Besides the global semantic similarity between clusters, SPICE [31] considers the extra semantic consistency of local samples to improve the pseudo-labeling accuracy. Although these methods work well on standard benchmarks, such as CIFAR-10 [21] and STL-10 [10], they still fail on real traffic sign datasets with a much larger quantity of classes and the serious class imbalance problem.

3 Proposed method

In this section, we detail the proposed detection framework and the automatic labeling pipeline.

3.1 Cascaded detection framework

The detection framework consists of a localization module and a recognition module, as shown in Fig. 1. The localization module predicts the center point coordinate, width, and height of the possible traffic signs. The patches are cropped from the original images and resized to 64 pixels square. The following recognition module predicts the class of each patch image. During the training stage, only the localization module needs the labeled dataset, while the recognition module uses an automatic labeling pipeline to generate the pseudo labels for each sample.

3.1.1 Localization module

Compared with the generic object detection task, traffic signs have a relatively consistent visual appearance and shapes. The actual obstacle for traffic sign detection is that the proportion ratio of the objects in whole images is very tiny. For example, the resolution of images in TT-100K benchmark is uniform 2000×2000, and the traffic signs are around 25 pixels width, as the area proportion is around 10^{-4}. Most generic object detection models employ an input resolution of around 640 pixels, which would cause too much information loss for TSD. Instead, we use 960×960 pixels as the input resolution, which reduces the computations properly and does not downsample the objects too much. For rectangular images, the long side is resized to 960, and the shorter side is resized to keep the height-width ratio.

Figure 2 is the network structure of the localization module. The whole network consists of three parts: backbone network, neck network, and head module. The network only contains regular convolutional, batch normalization, activation, upsampling, and concatenation operations without any other complex operations. The backbone is a straight CNN that acts as the feature extraction module. To enlarge the receptive field, the first layer is an unfold layer that collects the neighboring 4 pixels into one pixel by increasing the image channels. Each convolution layer is followed by convolution with stride 2 to squeeze the feature map size. The 5th, 7th, 9th, and 10th layers' outputs are collected as the input feature maps for the neck network.

The neck part extracts the intermediate representations and combines them to form the final features. The FPN is employed due to its simple and efficient structure. The three output features correspond to the small, medium, and large objects. The features of small objects are from the shallow layers of the backbone, and the features of the large objects are from the deeper layers. The strides of feature maps of the small, medium, large are 8, 16, and 32, respectively. Each output feature is attached with a projection network which is composed of two convolution layers. The final output channels of the projection networks are 6. As the strides for each feature map are 8, 16, and 32, and the input image is 960×960, the final output feature map size is $120 \times 120, 60 \times 60$, and 30×30, respectively.

Then the head module converts the convolutional outputs to corresponding bounding boxes, which is similar to YoloV3 [34]. Square grids are defined on the input image with the same stride as the output features. Each pixel in

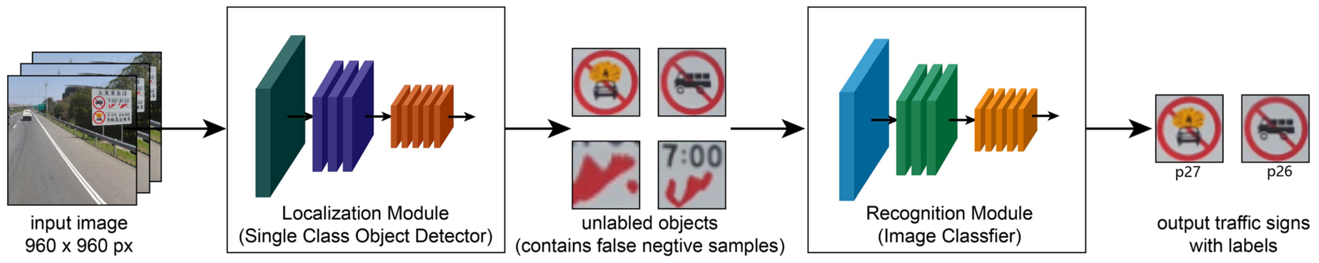


Fig. 1 The inference procedures of the proposed TSD approach. It consists of a localization module and a recognition module, which is a single class detector and an image classifier, respectively

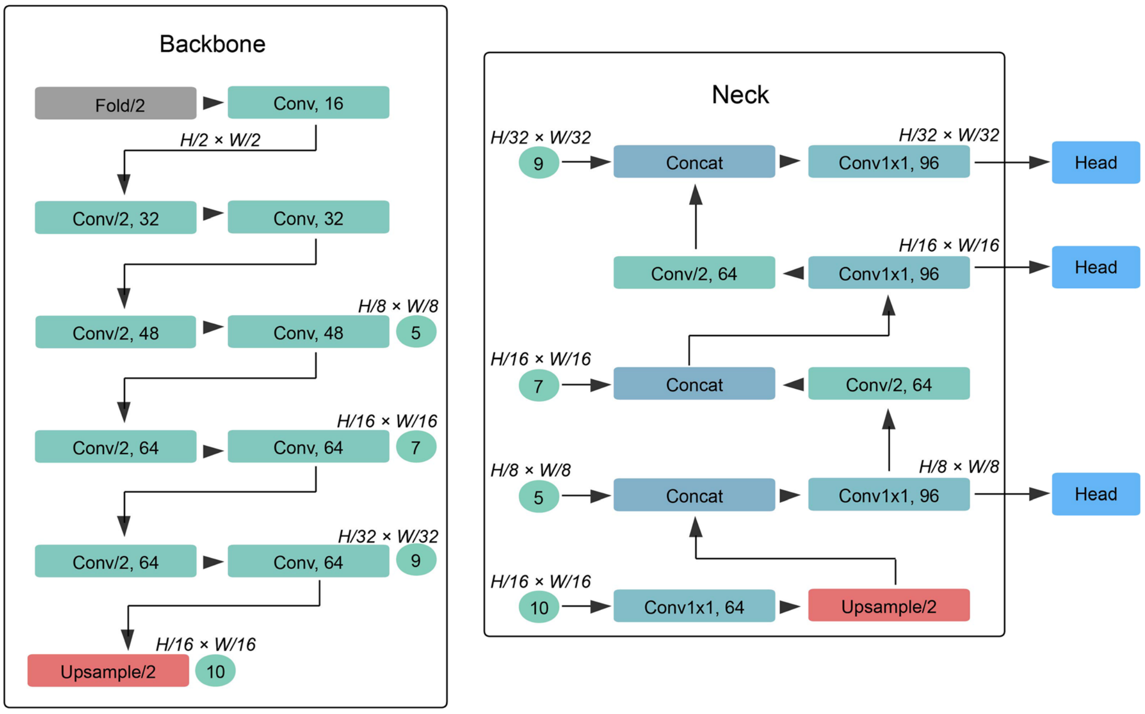


Fig. 2 The localization module network structure consists of three parts, including backbone, neck, and heads. The backbone is a straight CNN for extracting features from source images. The neck network combines features from the backbone in a hierarchical way. The head modules convert the features of the neck network to final boxes. “Conv” represents a block composed of convolution, batch norm, and activation. “Conv/2, N” means the stride is 2, and the kernel number is N

the output feature matches a point in the grid with a preset anchor box. Eqs. (1)–(4) are the predicted boxes where x, y, h, and w are the coordinates and shape of the predicted boxes. x_a, y_a, h_a, and w_a denote the preset anchor box's coordinates and shapes. x_t, y_t, h_t, and w_t correspond to 4 of the network output channels. During calculation of the regression loss, the x_t, y_t, h_t, and w_t are calculated reversely as the x, y, h, and w are set to be the ground truth bounding boxes. The extra two channels are employed to classify whether the anchor box contains an object.

$$x = w_a x_t + x_a \tag{1}$$

$$y = h_a y_t + y_a \tag{2}$$

$$w = w_a e^{w_t} \tag{3}$$

$$h = h_a e^{h_t} \tag{4}$$

3.1.2 Recognition module

With the predicted bounding boxes, we crop and resize the patches from the original input image to 64×64 pixels. Then the recognition module is actually an image classification network. We employ MobilNetV2-0.25 as the recognition module, which is a smaller version of MobileNetV2 [38]. MobileNetV2 is a common lightweight network that is composed of several inverted residual blocks and a fully connected layer for image classification. The separable depthwise convolution is the core computation layer. The network's width multiplier is set to 0.25, which makes the channel number a quarter of the original network. The maximum channel number of the network is 320, and the whole network only has 169K parameters. As the network width multiplier is 0.25 and the input resolution is set to 64, the FLOPs are approximately one percent of the original MobilenetV2. When tested on GPU, the computation latency of the recognition is around a quarter millisecond ignoring the data transmission time. Because the traffic sign usually has a significant visual feature, the traffic sign accuracy is still satisfied.

3.1.3 Training

The training procedure is split into two stages. The localization module is trained in the first stage. Then the localization module generates the proposed patches as the training samples for the recognition module. The labels of the training samples are generated from the proposed automatic labeling pipeline. Then the recognition module is trained in the second stage. In Sect. 4, we compare the detection accuracy of training with real labels and pseudo labels.

The loss for the localization module consists of two parts: the classification loss and the regression loss, which is similar to the Yolov3 [34]. The classification loss is the cross-entropy loss for classifying the anchor boxes while the regression loss is the L1 loss for adjusting the boundaries of the anchor boxes. We express the final localization loss as follows:

$$loss = \sum_{i}^{n}(w_i loss_{i1} + m_i loss_{i2}) \tag{5}$$

$$loss_{i1} = \hat{p_i}\log(p_i) + (1-\hat{p_i})\log(1-p_i) \tag{6}$$

$$loss_{i2} = \|y_i - \hat{y}_i\| \tag{7}$$

where i is the index of the anchor boxes, p_i is the predicted probability indicating the positive label, $\hat{p_i}$ is the ground truth label of the anchor box which is 0 or 1, m_i is the mask of the regression loss, and w_i is the assigned weight of the anchors. m_i is 0 if the IoU of the anchor box is below the threshold, and 1 otherwise. $\hat{y}_i$ is the ground truth vector of (x_t, y_t, w_t, h_t) in Eqs. (5)–(7). The y_i is the predicted values corresponding to the $\hat{y}_i$. $loss_{i1}$ and $loss_{i2}$ mean the classification loss and regression loss of an anchor, respectively.

The localization module is employed to propose the traffic sign candidates after its training procedure finished. The localization module predicts the possible traffic sign in each image, and the confidence threshold for the localization is set to a small value (0.01 for our experiments) for high recall rates. The generated candidates are the training samples of the recognition module. If the type of these candidates is accessible, we can use a fully supervised learning scheme to train the recognition. Otherwise, we can use the automatic labeling pipeline to label the candidates. The cross-entropy loss is employed to train the recognition module.

3.2 Automatic labeling pipeline

The goal of the automatic labeling pipeline is to find the pseudo labels of each unlabeled traffic sign. It provides the recognition module with the pseudo labels for training. The proposed pipeline consists of two parts as shown in Fig. 3.

3.2.1 Synthetic datasets

In the first stage, we create the synthetic datasets by augmenting the given traffic sign templates. An image classifier is trained on the synthetic datasets and then inference on the unlabeled dataset. Since the traffic signs are always painted on flat plates, the geometrical transformation of traffic sign appearance is relatively regular and tedious. The image augmentations are listed in Fig. 4, which can be categorized into two parts: geometrics augmentation and color augmentation. The geometric transformation includes translation, scaling, reflection, rotation, and shear mapping, which is able to simulate the real transformation well. We also employ the piecewise transformation to add the local distortion of the object. Then, the illumination, color, blur, noise, and dropout are applied to the images with random intensity. The final generated images are the random combinations of all these transformations. The generated object image is blended with the background, which is randomly cropped from the COCO [25] dataset. After the synthetic dataset is constructed, an image classifier is trained on the synthetic dataset for predicting the pseudo labels of the real samples.

A predicted label is adopted as the initial pseudo label if its confidence score is higher than its threshold. In our experiments, the threshold is 0.999 for TT-100K and GTSRB. For GTSDB, the threshold is 0.99. The reason to use two different thresholds is that the GTSDB only contains three classes, and 0.99 is enough for high precision. The wrong labeled samples have a serious effect on the model's performance. So, the threshold is a very high value here. Although the model trained on a synthetic dataset already can discriminate most of the samples, there are still many false pseudo labels. We infer that the gap between the real data and the artifact data causes the accuracy loss of the model. These incorrect labels can damage the subsequent training accuracy.

The background is collected from COCO dataset. Although there may be a few unlabeled traffic signs in background, the proposed approach is robust enough for

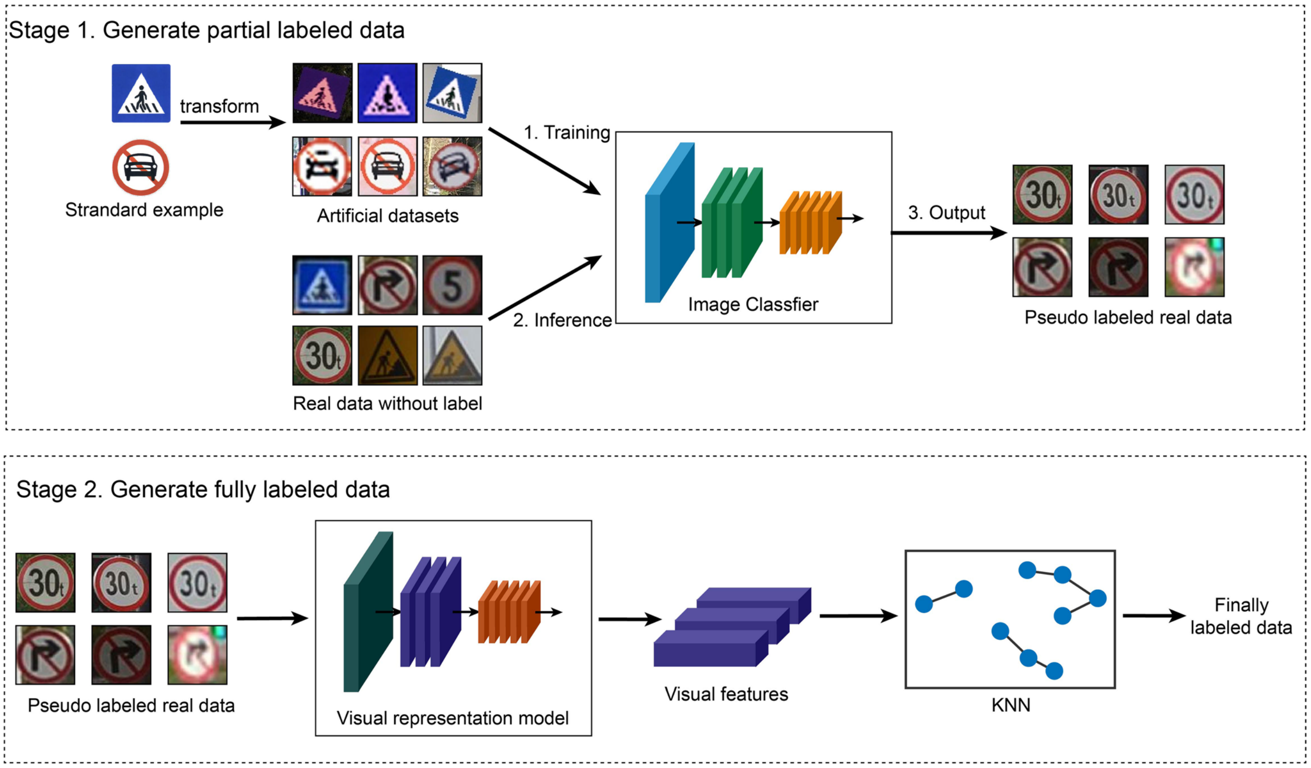


Fig. 3 The framework for automatic labeling consists of two stages. The first stage uses a synthetic dataset to train a model and generate labels. The second stage refines the labels by discarding unreliable samples

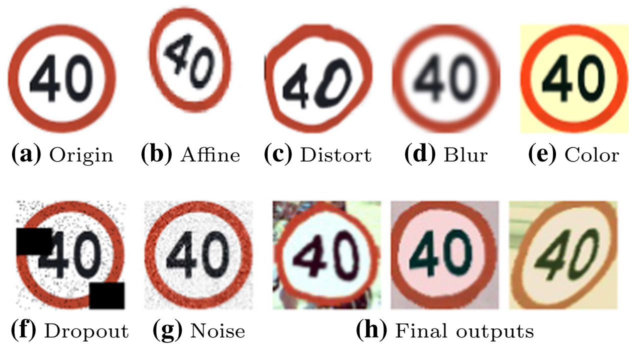


Fig. 4 The image augmentations are employed during the construction of the synthetic dataset. **a**–**g** Demonstrate the employed augmentations. **h** Three examples of the final combination augmentation

unlabeled traffic signs appearing in the background. Firstly, the main body of a sample in the synthetic data is a scheduled traffic sign. Even if there is an unlabeled traffic sign, the background only covers small area and its impact is very limited. Secondly, the synthetic data only works in stage one and does not participate in the rest procedures. The model trained by synthetic data is also only employed to generate the pseudo samples for the first round in stage two. The accuracy of this model is not good enough naturally. Thirdly, the number of traffic signs is quite small in COCO. Hence, the following procedures are robust to it, and the impact of unlabeled traffic signs in the background almost has no impact on the final performance.

3.2.2 Refine pseudo labels

In the second stage, we aim to reduce the false pseudo labels furthermore. Most incorrect labels are removed by thresholding in the previous stage. The remaining incorrect labels having high confidence scores are difficult to be identify. A representative visual representation model, SimCLR [8] is employed to obtain the similarity between each pair of the real samples. Based on contrastive learning, SimCLR does not need any labeled data. The training principle of SimCLR is represented in Fig. 5. $x_0, x_1, ...x_N$ are a batch of images randomly sampled from the training dataset, and each of them is augmented by pairs of random transformation, $t_i \sim \tau, i \in 1, 2, ...2N$. τ is the transformation sample distribution. The augmented images $\tilde{x_i}$ are projected into the latent space by the encoder network $f(\cdot)$. Then the projection network $h(\cdot)$ maps the latent vector h_i to the space where contrastive loss is applied. The projection network is an MLP network here. The contrastive loss function aims to identify the positive pairs from all augmented images. The loss function of a positive pair $\tilde{x_i}, \tilde{x_j}$ is defined as

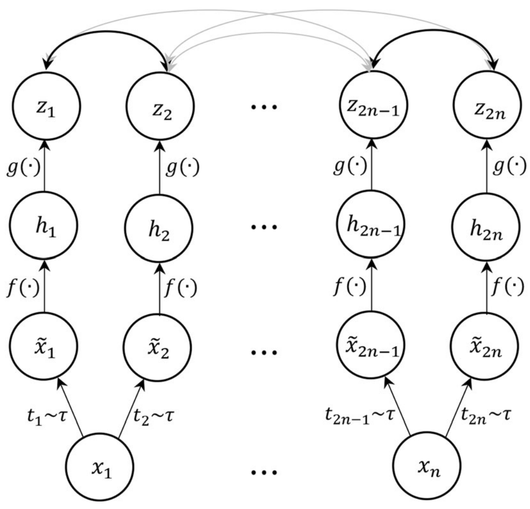


Fig. 5 The framework of the SimCLR algorithm. Two separate image augmentations are sampled from distribution τ and applied to obtain a pair of images. Encoder network $f(\cdot)$ and projection network $g(\cdot)$ are trained by contrastive loss. The visual representation $\boldsymbol{h}$ is employed for finding the nearest neighbor

$$l_{i,j} = \frac{exp(\text{sim}(z_i, z_j)/t)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/t)}, \tag{8}$$

where $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$, $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function that outputs 1 if k does not equal to i, and t denotes a temperature parameter. By comparing the latent vector of each sample, we identify the unreliable pseudo labels from the outputs of the first stage. To be specific, if a sample's nearest neighbor in latent space has a different pseudo label, we mark it as an unreliable sample. The unreliable samples are at high risk of having a false label and have serious effects on the model's accuracy.

To analyze the effectiveness of the refining pseudo labels, we make a comparison between models with or without the refining procedure. We evaluate the classification accuracy on the TT-100K dataset, and the result is represented in Table 1. The label recall is the proportion of the samples having the pseudo label. The refined label recall is lower than that of not refined, but the label accuracy can reach 100%. The accuracy of the model trained with refined labels increases 2%, which is a significant improvement.

Table 1 Ablation Study of refining procedure on the TT-100K dataset

	Label accuracy	Label recall	Model accuracy
Not refined	0.995	0.827	0.968
Refined	1.000	0.821	0.988

3.2.3 Iterations of label procedures

Once the unreliable pseudo labels are removed, the recognition module is trained on the initial pseudo labels. The process can be repeated several rounds to refine the pseudo labels successively. In the second round, the model generated from the first round can output more accurate labels, so the final pseudo label is better than before. In our experiments, we employ two rounds of pseudo label procedures and the accuracy plateaus in the third round. But for more complicated datasets, we may expect more rounds for higher accuracy.

4 Evaluation

To demonstrate the effectiveness of the proposed approach, we conduct evaluation experiments on three public benchmarks: TT-100K [55], GTSDB [19], GTSRB [40]. For the detection task, we follow the COCO evaluation metrics [25]. GTSRB is a traffic sign recognition dataset only for the evaluation of the recognition module.

4.1 Datasets

TT-100K [55] provides 1,000,000 images and 30,000 annotated traffic signs that are collected from Tencent Street Views in Chinese Cities. All the images are captured by high resolution SLR cameras on vehicles or shoulder-mounted equipment. By convention, 45 classes are adopted, and the left classes are ignored with fewer than 100 instances. The resolution of the provided images is uniform 2000 × 2000 pixels. Figure 6 is the size and aspect ratio distribution in TT-100K. During experiments, we filter the image without any traffic signs and left 6105 images for training and 3071 images for testing. Three classes noted as *io*, *wo*, and *po* are also ignored because they contain a mixture of several traffic signs.

GTSDB [19] provides 900 images and 1213 annotated traffic signs. All these images are collected from the video sequences recorded near Bochum, Germany. The final

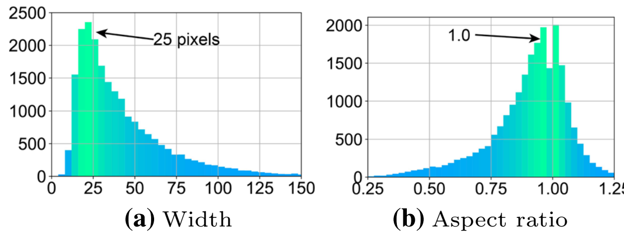


Fig. 6 The TT-100K data distribution. **a** and **b** The distribution of width and aspect ratio. The most frequent width and aspect ratio are 25 pixels and 1.0, respectively

images are clipped to 1366×800 pixels. The standard task for GTSDB focuses traffic signs and all classes are classified into four major categories: *prohibitory*, *danger*, *mandatory*, and *other*, while the *other* class is ignored during evaluation.

GTSRB [40] is a traffic sign recognition dataset, so we utilize it to evaluate the proposed recognition module. The images are collected from the same video sequences of GTSDB. It contains 43 classes and 51,839 images, which has 39,209 images for training and 12630 images for testing.

4.2 Training details

For the localization module, the optimizer is SGD with a base learning rate of 0.01. Weight decay is 0.005. The localization module is trained for 300 epochs. In the first three epochs, we use linear learning rate warming up schedule. And in the rest epochs, cosine learning rate decay is employed. The input image size is 960×960 for the localization module during training.

For the recognition module, we use AdamW [27] with an initial learning rate of 0.002 for the optimizer. The recognition module is trained for 100 epochs. The input image size is 64×64 for the recognition module. Weight decay is 0.01. The learning rate starts from 0.002 and is divided by 10 for every 30 epochs.

For SimCLR, the training data are the proposal candidates generated by the model in stage one. The optimizer is SGD with an initial learning rate of 0.4 and momentum of 0.9. Cosine learning rate decay is employed. The weight decay is 0.0001. The input image size is 64×64.

4.3 Detection performance

4.3.1 TT-100K

Because there are a limited number of public TSD models evaluated on TT-100K, we compare the proposed approach to other generic leading CNN-based detection approaches, including FasterRCNN [35], Yolov5L [50], PCN [24], and TT-100K-Benchmark. The experiment result is shown in Table 2. “Baseline” is the proposed model trained by the labels generated from stage one, which does not include refining procedure. “Ours-unsupervised” represents our detection framework trained with pseudo labels while “Ours-supervised” represents that with original labels. mAP@(N) is the mAP with IoU threshold N. mAP@[50:5:95] is the mathematical average value over mAP@50 to mAP@90 with stride 5. Readers can refer to [25] and their website for more details about mAP metrics.

FasterRCNN is implemented with the MMDetection [7]. Following the default setting, the network runs on the images resized to 1333 pixels and 800 pixels, which is larger than the proposed method. Yolov5L is the large model in YoloV5 model family and is implemented from the official code in our experiments. The input images are resized to 960 pixels which are the same as our approach. PCN model is also implemented from the official code. For a fair comparison, we use the same data as our method that excludes the “wo”, “io”, and “po” classes. The “TT-100K” indicates the baseline algorithm provided from the TT-100K benchmark.

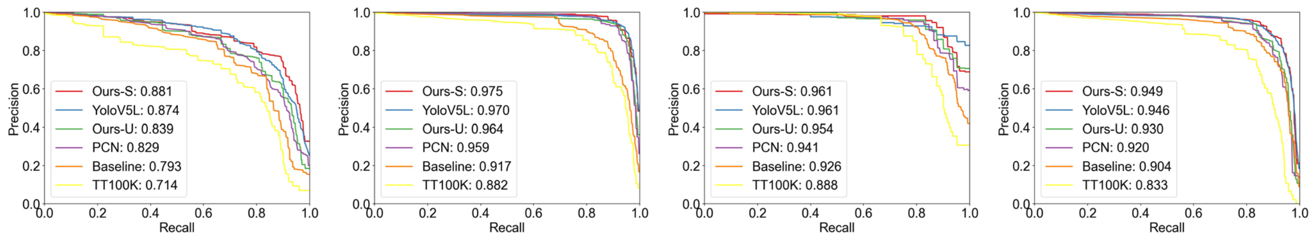
Figure 7 shows the precision-recall curves of mAP@50. We follows the TT-100K recommended evaluation scheme and divided the traffic-signs into three categories according to their size: small ($\text{area} < 32^2$ pixels), medium (32^2 pixels $< \text{area} < 96^2$ pixels), and large ($\text{area} > 96^2$ pixels). It shows that our detection framework outperforms others, and the model trained in pseudo labels only has a small accuracy degradation. Compared with the TT-100K and FasterRCNN, the proposed approach has a significant advantage on model size and accuracy. The proposed model only has the approximate 2% model size of the Yolov5L, but has a similar detection accuracy. The mAP50 of “Ours-supervised” is even higher than Yolov5L. The baseline model’s performance is lower than the model with refining procedures, and it proves the effectiveness of the proposed labeling pipeline.

4.3.2 GTSDB

Similar to the result of the TT-100K, the model trained with pseudo labels only has a slight accuracy drop. Table 3 shows the results of detection performance on GTSDB and other properties. The other model’s data are cited from [1]. For a fair comparison, our model’s frame per second (FPS) and memory consumption is tested with NVIDIA 2080Ti GPU using PyTorch framework [32] without extra optimization. The proposed model has the highest mAP and the smallest model size. The localization module contains 0.64 M parameters, and the recognition module contains 0.16 M parameters. The FLOPs and memory consumption is also very small compared with other methods. The memory usage is larger than SSD MobileNet, and the reason is that our model is a two-stage model. We stimulate 30 proposals for each input image, so the memory consumption is larger than single-stage models. The SSD MobileNet has the smallest FLOPs and fastest speed, but the mAP is unsatisfied. So, the proposed model has a better overall performance. The model trained with pseudo labels only has 0.6 descent of mAP on GTSRB.

Table 2 The comparison between mAPs on the TT-100K benchmark

	mAP@[50:5:95]	mAP@50	mAP@75	Parameters
TT-100K [55]	0.658	0.833	0.790	35.12 M
Faster-RCNN [35]	0.511	0.698	0.604	41.52 M
Yolov5L [50]	0.747	0.946	0.886	47.00 M
PCN [24]	0.721	0.920	0.866	34.35 M
Baseline	0.706	0.904	0.841	0.80 M
Ours-supervised	0.737	0.949	0.880	0.80 M
Ours-unsupervised	0.726	0.930	0.870	0.80 M



(a) small **(b)** medium **(c)** large **(d)** all

Fig. 7 The precision-recall curves on TT-100K dataset. The proposed method has the highest mAP@50 score. The model trained with the pseudo labels only has a small gap with the model trained with real labels

Table 3 Models' properties ordered by mAP on GTSDB Dataset

	mAP@50	Parameters (M)	FLOPs (G)	Memory (MB)	FPS
Faster R-CNN ResNet 50 [35]	91.52	43.34	533.58	5256	9.61
Faster R-CNN ResNet 101 [35]	95.08	62.38	625.78	6134	8.11
Faster R-CNN Inception V2 [35]	90.62	12.89	120.62	2175	2.26
R-FCN ResNet 101 [11]	95.15	64.59	269.90	3509	11.70
SSD MobileNet [26]	61.64	5.57	2.30	95	66.03
SSD Inception V2 [26]	66.10	13.47	7.59	284	42.12
YOLO V2 [33]	78.83	50.59	62.78	1318	46.55
Baseline	94.17	0.80	4.20	352	55.50
Ours-supervised	96.83	0.80	4.20	352	55.50
Ours-unsupervised	96.24	0.80	4.20	352	55.50

4.3.3 GTSRB

As a traffic sign recognition dataset, GTSRB is employed for evaluating the automatic labeling approach here. We repeat two rounds of labeling pipeline and the intermediate metrics of the pseudo labels are represented in Table 4. The model of the first round is trained with the synthetic dataset, and the second one is trained with the refined pseudo labels of the first round.

The recall of the pseudo labels is the labeled sample number divided by the total sample number. The accuracy of the pseudo labels is the correct rate in the labeled samples. The accuracy of the "all labels" in Table 4 is the global accuracy of all predicted labels, whether it is collected as a pseudo label, so recall metric is not applicable here.

Table 4 Accuracies of Pseudo Labels on Different Stages

Round	Round	Metrics
First	All labels	Accuracy=0.956
	Initial pseudo labels	Recall = 0.827 accuracy = 0.995
	Refined pseudo labels	Recall = 0.821 accuracy = 1.000
Second	All labels	Accuracy = 0.994
	Initial pseudo labels	Recall = 0.988 accuracy = 0.999
	Refined pseudo labels	Recall = 0.986 accuracy = 1.000

Regarding the first round, the label recall is not very high, but the refined pseudo labels have 100% accuracy. In the second round, the model trained on the labels generated by the first round achieves a very high recall and accuracy.

The accuracies of the refined pseudo labels are 100% in both two stages, indicating that the pseudo label refining method can remove the unreliable labels efficiently with negligible descent of recall rate. The final refined pseudo labels can cover approximately 99% samples with 100% accuracy.

The results in Table 5 provide a comparison of the classification accuracy of some state-of-the-art weakly/semi-supervised TSR methods and the proposed method on GTSRB dataset. It is essential to note that the proposed method only uses 43 template images to achieve the highest accuracy, while the other methods need much more labeled images for training. Although ROSSOT, MFF, and the proposed method both need to generate the pseudo labels for training in the next iteration, the proposed method provides a more robust scheme to reduce the wrong pseudo labels.

4.4 Speed performance

In Table 3, we only use the standard Pytorch framework for a fair speed comparison. In Fig. 8, we visualize the comparison between FPS, model size, and accuracy. The mAP@50 is computed on TT-100K dataset. Each circle area in Fig. 8 indicates the parameter number of each model. The proposed models are located in the upper right corner, which have high accuracy and speed. We also give the visualization of some detection results in Fig. 9.

To demonstrate the full performance of the proposed models, we evaluate the inference speed of the proposed model on the Nvidia RTX 2080Ti and the Nvidia Jetson Xavier by TensorRT [43] framework. The RTX 2080Ti is a high-performance GPU with powerful computation capacity, and the Jetson Xavier is a lightweight edge device with only 30 W power consumption. NVIDIA TensorRT is an SDK for high-performance deep learning inference, which is carefully optimized for network inference. For RTX 2080Ti, the CPU is a 10-core Intel i9-9900X @ 3.50 GHz. The RAM is DDR4 of 64G with a bandwidth of 42.6 GB/s. For Jetson Xavier, the CPU is an 8-core ARM v8.2. The RAM is 256-bit LPDDR4x of 32 GB with a bandwidth of 136.5GB/s.

We evaluate the latency and the average latency on both devices. The latency is the time consumed when batch size

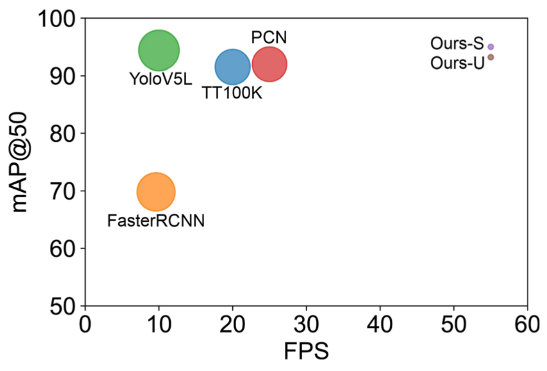


Fig. 8 The comparison between FPS, model size, and accuracy. Circle areas indicate the parameter numbers

is 1, while the average latency is the time spent on each image when batch size is 8. We simulate 30 proposal boxes in a single image. When the batch size is 8, the recognition module receives 240 proposal patches. The results are listed in Table 6.

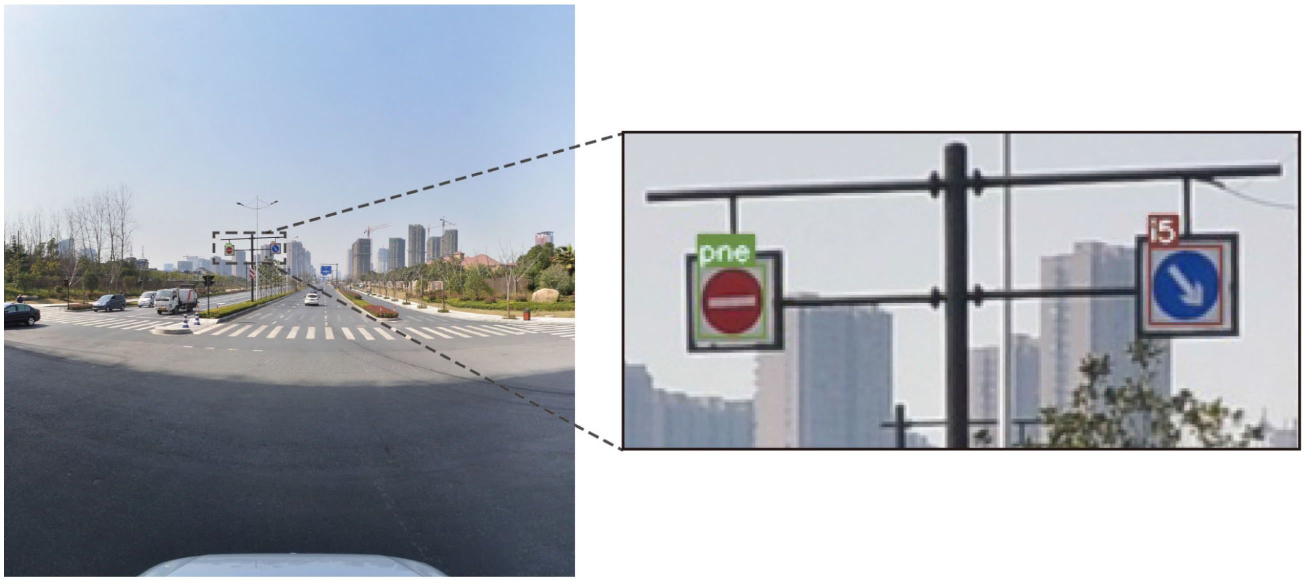
The total latency is mainly composed of three parts: the data transmission, the localization computation latency, and the recognition computation latency. On RTX 2080Ti, the total latency is 3.038 ms with single image batch data. Considerable time is spent by data transmission. The calculation latency is only 1.27 ms when batch size is 1, and it is smaller with batch size of 8. The average latency for batch size of 8 is smaller because the model is too small to utilize all resources when there is only one input image. Data transmission and calculation can be pipelined to increase the throughput in real applications.

On Jetson Xavier, the total latency is 8.266 ms for single image batch data. When the batch size is 8, the average computation latency is 5.831 ms. The data transmission time is shorter than that of GPU. We speculate that the reason is that the Jetson Xavier have a larger memory bandwidth.

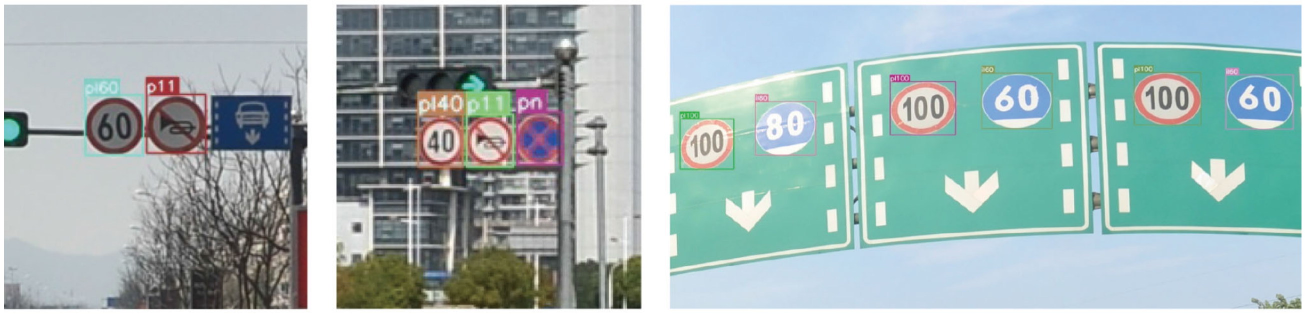
In conclusion, the proposed model has small latency far below the requirement whether on GPU or edge devices, making it suitable for various deployments.

Table 5 Compared with other weak/semi-supervised methods on GTSRB

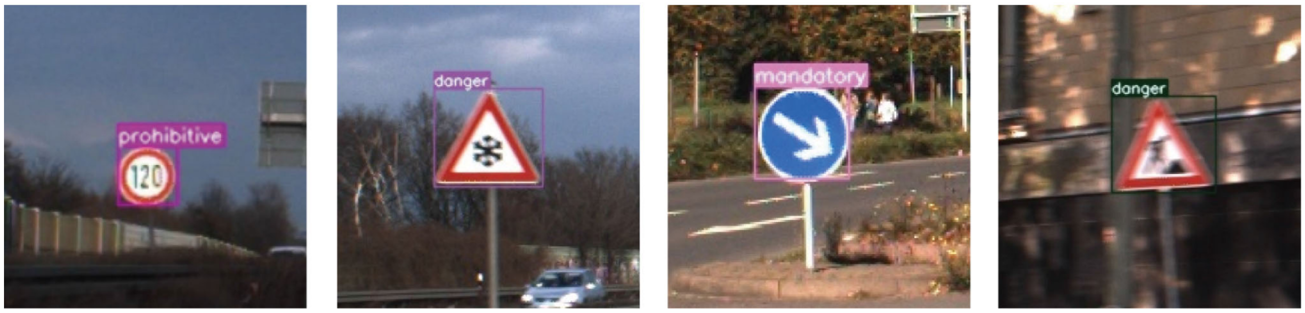
	Labeled data	Accuracy (%)
MFF [17]	1960 images (5% of training data)	95.05
MFF [17]	3920 images (10% of training data)	97.16
ROSSOT [30]	23,525 images (60% of training data)	99.27
Ours-U	43 template images	99.40



(a) The visualization of detection results on an original image.



(b) The demonstration of detection results on the TT-100K dataset.



(c) The demonstration of detection results on the GTSDB dataset. The images are cropped from the original images.

Fig. 9 The demonstration of detection results. **a** The proportion of objects in the image is very small. **b** and **c** The images are cropped from original images

Table 6 Latency (milliseconds) of the proposed model accelerated by TensorRT

	Batch size	Data	Localization	Recognition	Total
GPU	1	1.918	0.866	0.254	3.038
GPU	8	1.232	0.662	0.083	1.977
Jetson	1	1.237	5.902	1.127	8.266
Jetson	8	0.486	5.207	0.624	6.317

5 Conclusion

In this paper, we propose a lightweight detection model for traffic sign detection and an automatic labeling pipeline. Composed of a localization and a recognition module, the detection model is efficient and can achieve millisecond level detection speed on GPUs and edge devices. The automatic labeling pipeline reduces the manual labeling workload greatly and also achieves promising accuracy. The experiments demonstrate the comprehensive performance of the proposed method surpasses the state-of-the-art approaches. Since volatile traffic sign detection applications, the proposed method is meaningful in data preparation and deployments.

Funding This work was supported in part by the National Natural Science Foundation of China under Grants 62174084, 62104097 and in part by the High-Level Personnel Project of Jiangsu Province under Grant JSSCBS20210034, the Key Research Plan of Jiangsu Province of China under Grant BE2019003-4.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Arcos-Garcia A, Alvarez-Garcia JA, Soria-Morillo LM (2018) Evaluation of deep neural networks for traffic sign detection systems. Neurocomputing 316:332–344
2. Berkaya SK, Gunduz H, Ozsen O et al (2016) On circular traffic sign detection and recognition. Expert Syst Appl 48:67–75
3. Cao S, Zheng W, Mo S (2019) Unsupervised data augmentation for improving traffic sign recognition. In: Pacific Rim international conference on artificial intelligence, pp 297–306
4. Caron M, Bojanowski P, Joulin A et al (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European conference on computer vision (ECCV), pp 132–149
5. Caron M, Misra I, Mairal J et al (2020) Unsupervised learning of visual features by contrasting cluster assignments. In: Thirty-fourth conference on neural information processing systems (NeurIPS)
6. Chang J, Wang L, Meng G et al (2017) Deep adaptive image clustering. In: Proceedings of the IEEE international conference on computer vision, pp 5879–5887
7. Chen K, Wang J, Pang J et al (2019) MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:190607155
8. Chen T, Kornblith S, Norouzi M et al (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning, pp 1597–1607
9. Chowdhury SR, Tornberg L, Halvfordsson R et al (2019) Automated augmentation with reinforcement learning and gans for robust identification of traffic signs using front camera images. In: 2019 53rd Asilomar conference on signals, systems, and computers. IEEE, pp 79–83
10. Coates A, Ng A, Lee H (2011) An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 215–223
11. Dai J, Li Y, He K et al (2016) R-FCN: object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, pp 379–387
12. De La Escalera A, Moreno LE, Salichs MA et al (1997) Road traffic sign detection and classification. IEEE Trans Ind Electron 44(6):848–859
13. Fu CY, Liu W, Ranga A et al (2017) DSSD: deconvolutional single shot detector. arXiv preprint arXiv:170106659
14. Fujita H, Selamat A (2019) Investigation on data augmentation for object detection sing deep neural network for traffic signs application. In: Advancing technology industrialization through intelligent software methodologies, tools and techniques: proceedings of the 18th international conference on new trends in intelligent software methodologies, tools and techniques, p 144
15. Grill JB, Strub F, Altché F et al (2020) Bootstrap your own latent—a new approach to self-supervised learning. In: Advances in neural information processing systems, pp 21271–21284
16. He K, Fan H, Wu Y et al (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9729–9738
17. He Z, Nan F, Li X et al (2019) Traffic sign recognition by combining global and local features based on semi-supervised classification. IET Intell Transp Syst 14(5):323–330
18. Horn D, Houben S (2020) Fully automated traffic sign substitution in real-world images for large-scale data augmentation. In: 2020 IEEE intelligent vehicles symposium (IV). IEEE, pp 465–471
19. Houben S, Stallkamp J, Salmen J et al (2013) Detection of traffic signs in real-world images: The German traffic sign detection benchmark. In: The 2013 international joint conference on neural networks (IJCNN), pp 1–8
20. Jin J, Fu K, Zhang C (2014) Traffic sign recognition with hinge loss trained convolutional neural networks. IEEE Trans Intell Transp Syst 15(5):1991–2000
21. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images. Technical Report
22. Li F, Qiao H, Zhang B (2018) Discriminatively boosted image clustering with fully convolutional auto-encoders. Pattern Recognit 83:161–173
23. Li H, Sun F, Liu L et al (2015) A novel traffic sign detection method via color segmentation and robust shape matching. Neurocomputing 169:77–88
24. Liang Z, Shao J, Zhang D et al (2020) Traffic sign detection and recognition based on pyramidal convolutional networks. Neural Comput Appl 32(11):66
25. Lin TY, Maire M, Belongie S et al (2014) Microsoft COCO: common objects in context. In: European conference on computer vision, pp 740–755
26. Liu W, Anguelov D, Erhan D et al (2016) SSD: single shot multibox detector. In: European conference on computer vision, pp 21–37
27. Loshchilov I, Hutter F (2017) Decoupled weight decay regularization. arXiv preprint arXiv:171105101
28. Maldonado-Bascón S, Lafuente-Arroyo S, Gil-Jimenez P et al (2007) Road-sign detection and recognition based on support vector machines. IEEE Trans Intell Transp Syst 8(2):264–278
29. Mogelmose A, Trivedi MM, Moeslund TB (2012) Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans Intell Transp Syst 13(4):1484–1497

30. Nartey OT, Yang G, Asare SK et al (2020) Robust semi-supervised traffic sign recognition via self-training and weakly-supervised learning. Sensors 20(9):2684
31. Niu C, Wang G (2021) Spice: semantic pseudo-labeling for image clustering. arXiv preprint arXiv:210309382
32. PyTorch (2021) PyTorch. https://pytorch.org/
33. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
34. Redmon J, Farhadi A (2018) YoloV3: an incremental improvement. arXiv preprint arXiv:180402767
35. Ren S, He K, Girshick R et al (2015) Faster R-CNN: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99
36. Ruta A, Li Y, Liu X (2010) Real-time traffic sign recognition from video by class-specific discriminative features. Pattern Recognit 43(1):416–430
37. Salti S, Petrelli A, Tombari F et al (2015) Traffic sign detection via interest region extraction. Pattern Recognit 48(4):1039–1049
38. Sandler M, Howard A, Zhu M et al (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
39. Shen L, You L, Peng B et al (2021) Group multi-scale attention pyramid network for traffic sign detection. Neurocomputing 452:1–14
40. Stallkamp J, Schlipsing M, Salmen J et al (2011) The German traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks. IEEE, pp 1453–1460
41. Sun W, Du H, Nie S et al (2019) Traffic sign recognition method integrating multi-layer features and kernel extreme learning machine classifier. Comput Mater Contin 60(1):147–161
42. Tabernik D, Skočaj D (2019) Deep learning for large-scale traffic-sign detection and recognition. IEEE Trans Intell Transp Syst 21(4):1427–1440
43. TensorRT (2021) TensorRT. https://github.com/NVIDIA/TensorRT
44. Tian K, Zhou S, Guan J (2017) Deepcluster: a general clustering framework based on deep learning. In: Joint European conference on machine learning and knowledge discovery in databases, pp 809–825
45. Tian Y, Gelernter J, Wang X et al (2019) Traffic sign detection using a multi-scale recurrent attention network. IEEE Trans Intell Transp Syst 20(12):4466–4475
46. Van Gansbeke W, Vandenhende S, Georgoulis S et al (2020) Scan: learning to classify images without labels. In: European conference on computer vision, pp 268–285
47. Wu Y, Li Z, Chen Y et al (2020) Real-time traffic sign detection and classification towards real traffic scene. Multimedia Tools Appl 79(25):18,201-18,219
48. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International conference on machine learning, pp 478–487
49. Yang J, Parikh D, Batra D (2016) Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5147–5156
50. YoloV5 (2021) YoloV5. https://github.com/ultralytics/yolov5
51. Zhang J, Wang W, Lu C et al (2020) Lightweight deep network for traffic sign classification. Ann Telecommun 75(7):369–379
52. Zhang J, Xie Z, Sun J et al (2020) A cascaded r-CNN with multiscale attention and imbalanced samples for traffic sign detection. IEEE Access 8:29,742-29,754
53. Zhou P, Hou Y, Feng J (2018) Deep adversarial subspace clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1596–1604
54. Zhou S, Liang W, Li J et al (2018) Improved VGG model for road traffic sign recognition. Comput Mater Contin 57(1):11–24
55. Zhu Z, Liang D, Zhang S et al (2016) Traffic-sign detection and classification in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2110–2118